DOCUMENT RESUME

ED 038 839 EM 007 809

AUTHOR	Fremer, John; Anastasio, Ernest J.
TITLE	Computer-Assisted Item Writing--I. Spelling Items.
INSTITUTION	Educational Testing Service, Princeton, N.J.
REPORT NO	TDM-68-1
PUB DATE	Jul 68
NOTE	12p.
EDRS PRICE	EDRS Price MF-$0.25 HC-$0.70
DESCRIPTORS	Computer Assisted Instruction, *Computer Programs, Educational Technology, Item Analysis, *Programing, *Spelling Instruction, Testing

ABSTRACT

Computers have been used successfully in test assembly following the cooperation of test specialists and computer specialists in systematic analyses of the test-assembly process. Systematic exploration of the item-writing process may well make it possible for the computer to take on some of the less creative portions of item writing. The current study demonstrates the potential usefulness of the computer as a tool for an item writer. A spelling item type was used for this demonstration, as it seemed to have the fewest facets or dimensions. An analysis was then made of the types of misspellings which are used by writers of spelling items. A set of error-generation rules was developed and a computer program [The MISSPELLER] was written. A sample of words was fed into the computer and a list of misspelled words, separated into previously defined error categories was created. The list was then evaluated by spelling-test developers and judged to be a useful resource. Other more complex item types are now being studied. Appendices contain samples of the printouts obtained. (Author/JY)

TEST DEVELOPMENT MEMORANDUM
TDM-68-1 JULY 1968

Computer-Assisted Item Writing — I (Spelling Items)

John Fremer
Test Development Division, ETS
Ernest J. Anastasio
Developmental Research Division, ETS

EDUCATIONAL TESTING SERVICE
PRINCETON, NEW JERSEY
BERKELEY, CALIFORNIA

COMPUTER-ASSISTED ITEM WRITING — I (SPELLING ITEMS)

JOHN FREMER
Test Development Division, ETS

ERNEST J. ANASTASIO
Developmental Research Division, ETS

Test Development Memorandum
TDM-68-1
July 1968

EDUCATIONAL TESTING SERVICE
Princeton, New Jersey

Abstract

Computers have been used successfully in test assembly following the cooperation of test specialists and computer specialists in systematic analyses of the test-assembly process. Systematic exploration of the item-writing process may well make it possible for the computer to take on some of the less creative portions of item writing. The current study demonstrates the potential usefulness of the computer as a tool for an item writer. A spelling item type was used for this demonstration, as it seemed to have the fewest facets or dimensions. An analysis was then made of the types of misspellings which are used by writers of spelling items. A set of error-generation rules was developed and a computer program [The MISSPELLER] was written. A sample of words was fed into the computer and a list of misspelled words, separated into previously defined error categories was created. The list was then evaluated by spelling-test developers and judged to be a useful resource. Other more complex item types are now being studied.

Computer Assisted Item Writing —I (Spelling Items)[1]

General Introduction

The widespread availability of computers is causing farsighted individuals in all areas to consider new ways to employ this valuable tool. The first step which a potential computer-user must take is that of systematically analyzing the tasks which he sees as amenable to computer assistance. At Educational Testing Service a program is now in progress to turn over to a computer a major part of the task of assembling items into tests. Careful study of the logic of test assembly necessarily preceded the development of appropriate computer programs. Once experienced test developers and computer specialists learned to speak each other's language it was possible to work toward a mutually comprehensible description of the steps involved in test assembly.

A pilot study by Rock (1965) indicated that at that time the computer could produce a test comparable to one assembled by a relatively inexperienced human test developer.

The art of computer assembly has progressed considerably since Rock's study. Given a pool of classified items, the computer can now produce operational tests that meet previously defined content and statistical specifications.

The research reported here represents the first part of a project which aims to explore the item-writing process, with a view toward developing computer techniques that would make possible the generation of verbal items. Our objectives include the identification of the properties of words and sentences that are most relevant to the creation of verbal items and the development of rules for the coding of words and sentences so that they can be manipulated by the computer.

---

[1]Paper presented at the annual meeting of the National Council on Measurement in Education, Chicago, Illinois, February, 1968.

There have been a number of recent indications that the time has arrived to employ the computer as an item writer. Osburn (1966), for example, has developed a computer procedure for writing statistics problems. He uses a series of "question frames" containing fixed and variable components. Each time a change is made in a variable component, a new item is created. Some resemblance can be seen between Osburn's approach and that used by Guttman (1965) in his facet analysis.

Another bit of evidence for the feasibility of computer assistance in item writing is Richards' recent study, "Can Computers Write College Admissions Tests?" (Richards 1967). Richards constructed a synonym test by simulating a computer, using Roget's Thesaurus as a word base. He subsequently tested entering freshman at the University of Iowa and found that his 72-item test was less reliable than the 48-item Wide Range Vocabulary Test but that it was equally effective as a predictor of first-semester Grade Point Average.

The simulation study by Richards bypassed such problems as creating a data base and writing the actual computer programs, but in the words of its author: "The simulation was rigorous...and the items correspond exactly to what would be written by a computer."

An Outline For an Exploratory Study

Richard's study stimulated the interest of some of the individuals at Educational Testing Service engaged in developing verbal tests and of some researchers who were generally interested in computer applications to education. A series of meetings on possible ways to follow up this shared interest led to plans for exploratory studies of computer applications to a variety of verbal-item types. The first item types to be considered were those involving only single words instead of items requiring a larger context. A decision was finally made to focus initially on the simplest single-word item type—the spelling item.

The spelling item seemed a particularly desirable starting point for a limited demonstration of computer potential, as the relevant item dimensions seemed to be easily defined. The particular type of spelling item which was considered presents a student with four incorrectly spelled words and one correctly spelled word. A sample item with directions follows:

Directions: From each group of words, select the word that is correctly spelled and blacken the corresponding space on your answer sheet.

There is one and only one correctly spelled word in each group.

Example:

(A) imerse
(B) decide
(C) numm
(D) eihgt
(E) sould

It was first necessary to agree on what would constitute a minimal demonstration of the computer's item-writing potential. We expected to find the task of generating items to be more difficult than the previously explored task of assembling a test from an existing item pool. We therefore decided to make "computer assistance" our first goal and to attempt to produce a pool of misspelled words to serve as a resource file for an item writer, rather than to attempt to produce complete items. We agreed that the following five steps would be necessary:

1. Develop rules for generating misspelled words
2. Write a computer program for applying the rules to words
3. Select a group of words to serve as a data base
4. Run the program
5. Evaluate the computer output

Developing Error-Generation Rules

The task of writing out rules for generating plausible misspellings was carried out initially by research personnel who had no previous experience with the construction of spelling items. A list of misspelled words previously used in spelling tests was supplied as a reference document for discovering error-generation rules or algorithms. The task of developing algorithms proved quite simple, requiring but 3 to 4 hours of time from two psychologists who had never before "consciously attempted" to produce spelling errors. A total of forty rules were developed

initially. Here are two such rules stated in sentence form:

> If the letter-pair i e appears in a word, replace this letter-pair with the letter-pair e i.
>
> If a doubled consonant appears in a word, replace this doubled consonant with a single consonant.

The list of error rules was examined by staff members who had worked on spelling tests. They expressed considerable surprise both at the number of rules and at the fact that these rules seemed capable of generating many plausible misspellings. The spelling-test writers soon discovered, however, that they could produce additional error rules, based on their own item writing habits or on available information on frequent misspellings. It became clear that we would be able to produce enough misspellings to bring tears to the eyes of a spelling teacher. We faced the problem, on the other hand, that many of the rules produced implausible misspellings when applied to all words. Separating the rules according to the part of the word to which they should be applied promised to help somewhat, but it was decided that the ratio of plausible to implausible misspellings might best be improved by first creating a pool of misspellings and then noting which misspellings were selected for items by item writers and later chosen as correct by test takers.

Programming

Programming of the rules was done in SNOBOL, a language developed at Bell Telephone Laboratories especially to facilitate the handling of strings of symbols or characters by a computer. This language is most appropriate for tasks that do not require extended arithmetic computations. Researchers who would like to make greater use of computers, but are discouraged by the formidable task of learning a specialized and abstract computer language, should investigate the possibility of using SNOBOL. This flexible high-level computer language has two characteristics which increase its accessibility to a potential user:

1. The user can describe procedures in a nontechnical language which resembles natural language in some respects.

2. The user can write a complete program without involving himself in the complexities of machine operations.

For the current study SNOBOL offered a distinct advantage: a complete operating program could be written in approximately six man-hours. Also, the program is in a form which allows non-computer people to understand the structure and selection logic with relatively little guidance. A partial listing of the program[2] used for creating spelling errors is reproduced as Appendix 1. This listing includes only a sample of the error rules which were eventually developed.

The program generates misspellings by applying the error-generation rules to words. A word is defined as a string of characters which is preceded and followed by a blank space. The error-generation rules have been coded as a series of replacement functions with the previously described form—"If you see this, replace it with that." In attempting to apply the error-generation rules, the misspeller program does pattern matching. That is, it searches a word or string of characters for particular substrings or ordered sequences of characters. The patterns to be matched are literal character combinations, that is, specific letter combinations, and are represented as the first term in each of the replacement rules. When the program succeeds in matching a pattern, that pattern is replaced by the second term of the replacement rule.

Rules for the beginning or end of a word are applied to the first "n" or last "n" characters of a word, where "n" is equal to the number of characters in the first term of the replacement rule. Appendix 2 is a sample of the misspeller program output. The numbers alongside the misspellings refer to the error-generation rules contained in Appendix 1. The first misspelling for the word ACQUIESCENCE was generated by the application of rule number five. In attempting to apply this rule, the computer looked only at the first three letters of every word, since the pattern

---

[2]The authors gratefully acknowledge the assistance of Carl Helm in the preparation of the computer program.

"ACQ" that it was trying to match has only three letters. In attempting to apply each of its replacement rules in succession to the middle of a word, the computer starts with the second character of the word and proceeds through it. When a rule fails, that is, when the pattern is not matched any place in a word, the program simply advances to the next replacement rule. When a pattern is matched, however, the misspelled word is written into an output file and the rule is applied to the remainder of the word. This procedure permits a rule to be applied more than once to a particular word if a pattern that will trigger an error rule occurs more than once in that word. It should be noted, however, that each word in the list of misspelled words will contain only one error.

Summary and Conclusion

Our demonstration of the computer's ability to assist in the writing of spelling items has satisfied our original criteria. We have developed rules, programmed the rules, selected some words to be manipulated, run the program, and obtained an evaluation of the output by spelling-test item writers. The item writers indeed felt that lists of words with misspellings would be a helpful resource even with a large proportion of implausible misspellings. We have not, on the other hand, run the program on a large group of words and have not attempted to increase the efficiency of the program in terms of proportion of plausible misspellings. There may be enough information on student misspellings already available so that there is no practical reason for pursuing our spelling effort further. Even if this is true, we may eventually develop a compendium of words and their misspellings as an adjunct to some study of the spelling process and of the amount of confusion caused by misspellings.

For the present, however, we are turning our attention to other, more complex, and consequently more difficult-to-handle item types. We have so far begun the study of two additonal item types. Don Marcotte, a 1967 ETS summer graduate student, applied a syntactical analysis to sentences used in sentence completion

items in an attempt to identify structural communalities. Another study is now under way to develop a facet model of the various types of relationships employed in analogy items.

References

Guttman, L. A Faceted Definition of Intelligence. Scripta Hierosolymitane, 1965, 4, 166-181.

Osburn, H. G. Computer Aided Item Sampling for Achievement Testing. Paper read at the annual meeting of the American Psychological Association, New York, September 1966.

Richards, J. M. Jr. Can Computers Write College Admissions Tests? Journal of Applied Psychology, 1967, 51, 211-215.

Rock, D. Assembly of Tests by use of an Automated Item File. Research Memorandum 65-14, Princeton, N.J.: Educational Testing Service, 1965.

APPENDIX 1

RULES FOR MISSPELLING WORDS (THE MISSPELLER)

```
*RULES FOR BEGINNING OF WORD ONLY
      POINTER = '0'
    1. REPLACE('PRE','PER')
    2. REPLACE('PER','PRE')
    3. REPLACE('ADJ','AJ')
    4. REPLACE('AJ','ADJ')
    5. REPLACE('ACQ','AQ')
    6. REPLACE('AQ','ACQ')
*RULES FOR EVERYWHERE EXCEPT LEFT
    AND RIGHT TERMINALS

          LOOPB

      POINTER =POINTER + '1'
      WORD */'1'* *WORD*
    7. REPLACE('IE','EI')
    8. REPLACE('EI','IE')
    9. REPLACE('MB','MM')
   10. REPLACE('SH','S')
   11. REPLACE('CH','C')
   12. REPLACE('SC','S')
   13. REPLACE('SC','C')
   14. REPLACE('EA','EE')
   15. REPLACE('EE','EA')
   16. REPLACE('OR','ER')
   17. REPLACE('ER','OR')
   18. REPLACE('XH','H')
   19. REPLACE('PH','F')
   20. REPLACE('WR','R')
   21. REPLACE('GHT','HGT')
   22. REPLACE('ANC','ENC')
   23. REPLACE('ENC','ANC')
   24. REPLACE('AGE','EGE')
   25. REPLACE('EGE','AGE')
   26. REPLACE('ANT','ENT')
   27. REPLACE('ENT','ANT')
   28. REPLACE('ABL','EBL')
   29. REPLACE('EBL','ABL')
   30. REPLACE('ABL','IBL')
   31. REPLACE('IBL','ABL')
   32. REPLACE('EBL','IBL')
   33. REPLACE('IBL','EBL')
   34. REPLACE('ION','IN')
   35. REPLACE('OUL','OL')
   36. REPLACE('OUL','UL')
   37. REPLACE('PHY','PY')
   38. REPLACE('EUR','URE')
   39. REPLACE('CHO','CO')
   40. REPLACE('PRE','PER')
   41. REPLACE('PER','PRE')
   42. REPLACE('IZE','ISE')
   43. REPLACE('ISE','IZE')
   44. REPLACE('EDGE','EGE')
   45. REPLACE('EGE','EDGE')
   46. REPLACE('TION','SION')
   47. REPLACE('SION','TION')
   48. REPLACE('THYM','THEM')
   49. REPLACE('THYM','THM')
   50. REPLACE('S','C')
   51. REPLACE('C','S')
   52. REPLACE('Y','I')
   53. REPLACE('I','Y')
   54. WORD *A/'1'* *B/'1'*/F(LOOPC)
       EQUALS(A,B) REPLACE((A B),A)
            /(LOOPB)
```

RULES FOR TERMINALS ONLY

```
          LOOPC

      POINTER = LENGTH - '3'
      SAVE */POINTER* *WORD*
   55. REPLACE('ARY','RY')
   56. REPLACE('ERY','RY')
   57. REPLACE('ORY','RY')
   58. REPLACE('IRY','RY')
   59. REPLACE('ALY','LY')
   60. REPLACE('ELY','LY')
   61. REPLACE('ILY','LY')
      POINTER = POINTER + '1'
      SAVE */POINTER* *WORD*
   62. REPLACE('EL','LE')
   63. REPLACE('LE','EL')
   64. REPLACE('UL','ULL')
   65. REPLACE('EL','AL')
   66. REPLACE('AL','EL')
   67. REPLACE('AL','LE')
   68. REPLACE('LE','AL')
   69. REPLACE('EY','Y')
      POINTER = POINTER + '1'
      SAVE */POINTER* *WORD*
   70. REPLACE('Y','EY')
   71. REPLACE('I','Y')
```

APPENDIX 2

Sample Computer Output

	INITIAL	MIDDLE	TERMINAL
ACQUIESCENCE	5. AQUIESCENCE	7. ACQUEISCENCE	
		12. ACQUIESENCE	
		13. ACQUIECENCE	
		23. ACQUIESCANCE	
		50. ACQUIECCENCE	
		51. ASQUIESCENCE	
		51. ACQUIESSENCE	
		51. ACQUIESCENSE	
		53. ACQUYESCENCE	
EXCELLENT		27. EXCELLANT	
		51. EXSELLENT	
		54. EXCELENT	
MODIFIED		7. MODIFEID	
		53. MODYFIED	
		53. MODIFYED	
PREFERABLE	1. PERFERABLE	17. PREFORABLE	63. PREFERABEL
		28. PREFEREBLE	68. PREFERABAL
		30. PREFERIBLE	
REVEILLE		7. REVIELLE	63. REVIELEL
		53. REVEYLLE	68. REVIELAL
		54. REVEILE	
UNIVERSITY		17. UNIVORSITY	70. UNIVERSITEY
		50. UNIVERCITY	
		52. UNIVERSITI	
		53. UNYVERSITY	
		53. UNIVERSYTY	

ᵕ = Inversion

♦ = Omission

_ = Substitution

♦ = Insertion